

条件分岐と繰り返し

条件分岐 --- 場合分けを使った計算

```
irb(main):003:0> load("./ max.rb")
```

```
=> true
```

```
irb(main):004:0> max(123, 456)
```

```
=> 456
```

```
irb(main):005:0> max(max(12, 34), max(56, 78))
```

```
=> 78
```

```
def max(x,y)
  if y < x
    x
  else
    y
  end
end
max.rb
```

3 通りの場合分け

```
def sign(x)
  if x < 0
    -1
  else
    if 0 < x
      1      # not(x<0) and 0<x
    else
      0      # not(x<0) and not(0<x)
    end
  end
end
end
```

sign.rb

複雑な条件

- 色々な比較

書き方	数学	意味
$x > y$	$>$	x が y より大きい
$x \geq y$	\geq	x が y 以上
$x == y$	$=$	x と y が等しい (x=y でないことに注意)
$x < y$	$<$	x が y より小さい
$x \leq y$	\leq	x が y 以下
$x != y$	\neq	x と y が異なる

- 条件式の組合せ

書き方	意味
$x > y \ \ x == 0$	x > y <u>または</u> x == 0
$x < y \ \&\& \ y < z$	x < y <u>かつ</u> y < z
$!(x < y \ \&\& \ y < z)$	(x < y <u>かつ</u> y < z) <u>でない</u>

複雑な条件

= でないことに注意
= は代入

- 色々な比較

書き方	数学	意味
$x > y$	$>$	x が y より大きい
$x \geq y$	\geq	x が y 以上
$x == y$	$=$	x と y が等しい (x=y でないことに注意)
$x < y$	$<$	x が y より小さい
$x \leq y$	\leq	x が y 以下
$x != y$	\neq	x と y が異なる

- 条件式の組合せ

書き方	意味
$x > y \ \ x == 0$	x > y <u>または</u> x == 0
$x < y \ \&\& \ y < z$	x < y <u>かつ</u> y < z
$!(x < y \ \&\& \ y < z)$	(x < y <u>かつ</u> y < z) <u>でない</u>

どれが正しいか？

x の値が 7、y の値が 5、z の値が 3 であるとして

1. $x < y$
2. $z == y$
3. $z <= x$

どれが正しいか？

x の値が 7、y の値が 5、z の値が 3 であるとして

1. $x < y$
2. $x \leq y$
3. $y \neq z$
4. $z > x$
5. $z == x$

どれが正しいか？

x の値が 7、y の値が 5、z の値が 3 であるとして

1. $x > y \ \&\& \ z == x$
2. $x > y \ || \ z == x$
3. $!(x > y)$

どれが正しいか？

x の値が 7、y の値が 5、z の値が 3 であるとして

1. $x < y$
2. $x \leq y$
3. $x < y \ \&\& \ y \neq z$
4. $x \leq y \ \|\| \ y == z$
5. $!(x < y \ \&\& \ y == z)$

練習3.3a)c)

実数

- a) 2次方程式 $ax^2 + bx + c = 0$ の解の個数を求める `solutions(a,b,c)`. 判別式の値だけでなく、1次方程式になっている場合にも対応せよ。(quadratic.rb に定義を追加せよ。)
- c) 3つの異なる値 x, y, z が与えられたときの中央値を求める `median(x,y,z)`. 中央値とは大きさ順に並べたときに真ん中に来る値のことである。(median.rb というファイルを作れ。)

進捗状況の確認

1. 両方できた(できた時点で投票してください)
2. solutions だけ
3. median だけ
4. まだ何もできていない

真偽値を与える論理演算

irb(main):003:0> $x = 3$

=> 3

irb(main):004:0> $1 < x$

=> true

irb(main):005:0> $x == 2$

=> false

```
def is_even(x)
```

```
  x%2 == 0
```

```
end
```

is_even.rb

```
load("./is_even.rb")
```

```
def tnpo(n)
```

```
  if is_even(n)
```

```
    n/2
```

```
  else
```

```
    3*n + 1
```

```
  end
```

```
end
```

tnpo.rb

文字列

```
irb(main):003:0> s = "abra"
```

```
=> "abra"
```

```
irb(main):004:0> t = "cadabra"
```

```
=> "cadabra"
```

```
irb(main):006:0> u = s + t
```

```
=> "abracadabra"
```

```
irb(main):007:0> "123" + "456"
```

```
=> "123456"
```

```
irb(main):009:0> s.length()
```

```
=> 4
```

```
irb(main):010:0> (s + t).length()
```

```
=> 11
```

```
irb(main):012:0> s[0..0]
```

```
=> "a"
```

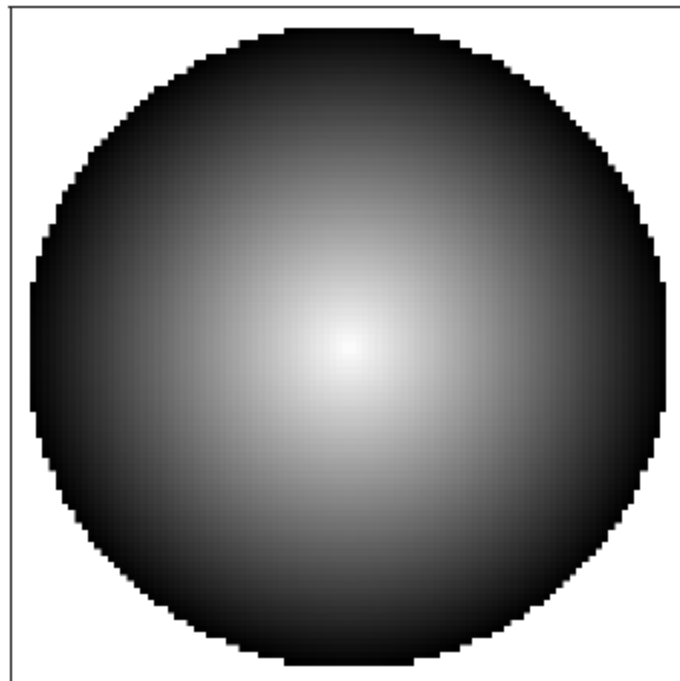
```
irb(main):013:0> s[1..2]
```

```
=> "br"
```

```
irb(main):014:0> t[1..(t.length()-1)]
```

```
=> "adabra"
```

繰り返しによる画像の作成



与えられた大きさの 1 次元配列を作る

```
>> image = Array.new(6)
```

```
=> [nil , nil , nil , nil , nil , nil]
```

```
>> a = Array.new(3)
```

```
=> [nil , nil , nil]
```

繰り返しによって、 配列の全要素をそれぞれ変更する

```
>> for i in 0..2
```

```
>>   a[i] = 0
```

```
>> end
```

```
=> 0..2
```

```
>> a
```

```
=> [0, 0, 0]
```

練習 (make1d)

- 大きさ n で中身が全て 0 であるような 1 次元配列を作る関数 `make1d(n)` を定義せよ。
(`make1d.rb` というファイルに格納する。)

```
def make1d(n)
  a = Array.new(?)
  for i in 0..?
    a[?] = ?
  end
  a
end
```

変数 `a` に入っている配列が
関数の値として返される

2次元配列を作る

```
>> for i in 0..5
```


```
>>   image[i] = make1d(3)
```

```
>> end
```

```
=> 0..5
```

```
>> image
```

```
=> [[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0,  
0, 0], [0, 0, 0]]
```



6行3列の配列

練習 (make2d)

- h 行 w 列の配列を作る `make2d(h,w)` を定義せよ。ただし、作られる配列の中身は全て 0 にせよ。(もちろん、`make1d` を使う。`make2d.rb` というファイルに格納する。)
 - `Array.new` によって配列を作るのを忘れずに。
 - 最後に配列を関数の値として返すのを忘れずに。

2重の繰り返し

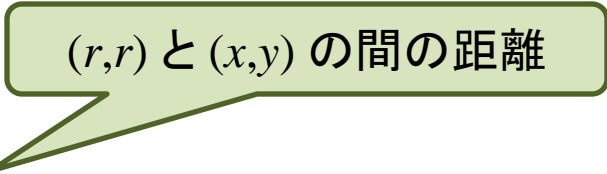
```
def sphere(s)
  image = make2d(s, s)
  for y in 0..(s-1)
    for x in 0..(s-1)
      image[y][x] = b(s, x, y)
    end
  end
end
image
end
```

変数 image に入っている配列が
関数の値として返される

練習3.2

- (sphere.rb をダウンロード。)
- 式(3.2)の計算をする関数 $b(s,x,y)$ を定義せよ。
(sphere.rb の中で定義すればよい。)

$$b(s, x, y) = \begin{cases} \frac{r - d(x, y, r, r)}{r} & (d(x, y, r, r) \leq r) \\ 1 & (d(x, y, r, r) > r) \end{cases} \quad (3.2)$$



(ただし $r = s/2.0$)

- show(sphere(100)) を実行して表示される画像を確かめよ。

進捗状況の確認

1. `show(sphere(100))` がうまく行った時点で投票してください。
2. `show(sphere(100))` がうまく行かない。
3. `b` が定義できた
4. `make2d` が定義できた
5. `make1d` が定義できた
6. まだ

make1d と make2d

- ここで、まだ make1d と make2d ができていない人は、配布プログラム make1d.rb と make2d.rb をダウンロードせよ (p.41)。
- 教科書 pp.60-61 を参照。

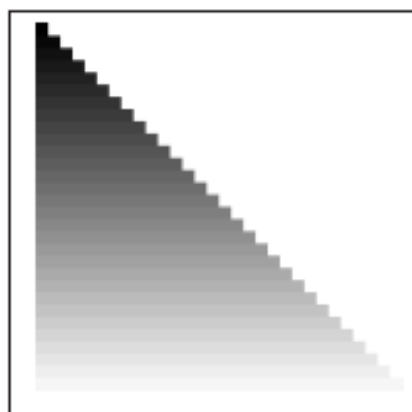
画面のファイルへの保存

- アプリケーションのユーティリティにある「グラブ」を起動する。
- メニューバーの「取り込み」メニューの「スクリーン」を選択し、指示に従って操作する。
- メニューバーの「ファイル」メニューの「保存」を選択する。ファイル名を聞かれるのでファイル名を入力する。特に指定しないと、「書類」のフォルダに保存されるので注意。
- レポートのメールにファイルを添付する。
 - 「Windows 対応の添付ファイルを送信」にチェックが入っていることを確認せよ。

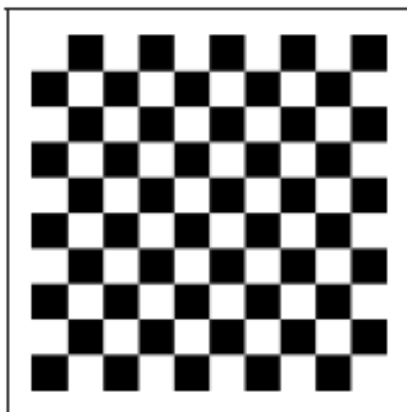
クラブできたか？

1. できた
2. まだ

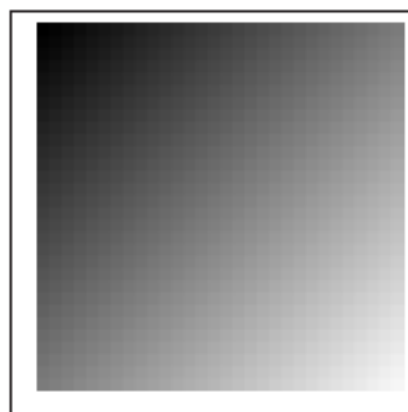
練習 26: (色々な図形*) 次のような画像を作成する関数をそれぞれ定義せよ。



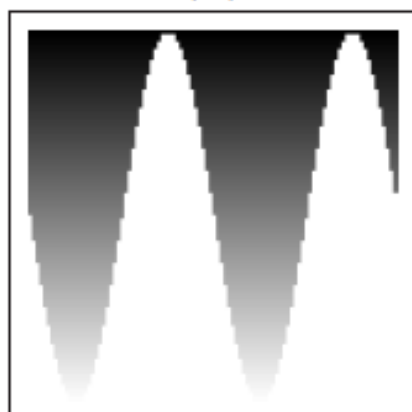
(a)



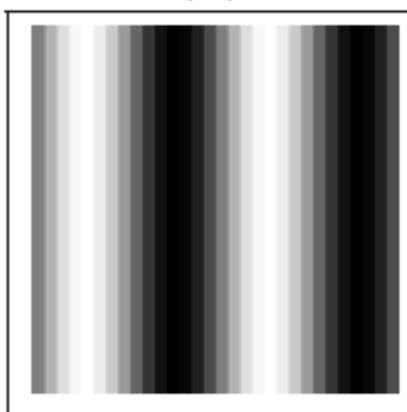
(b)



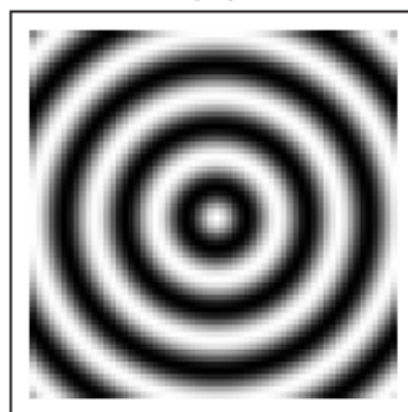
(c)



(d)



(e)



(f)

3.5 定義のまとめ

条件分岐: $\boxed{\text{式}_1}$ が成り立つときは $\boxed{\text{式}_2}$ を、そうでないときは $\boxed{\text{式}_3}$ を計算する条件分岐は次のように書く。このとき $\boxed{\text{式}_1}$ を条件式という。

```
if  $\boxed{\text{式}_1}$   
   $\boxed{\text{式}_2}$   
else  
   $\boxed{\text{式}_3}$   
end
```

真偽値: true と false はそれぞれ、真と偽を表わす値である。

文字列を作る: 2つの " 記号の間にある文字や記号は文字列を作る。

文字列の結合: $\boxed{\text{式}_1} + \boxed{\text{式}_2}$ という式は、 $\boxed{\text{式}_1}$, $\boxed{\text{式}_2}$ の値が文字列のとき、それらの文字列をつなげた文字列を作る。

文字列の長さ: $\boxed{\text{式}_1}.\text{length}()$ という式は、 $\boxed{\text{式}_1}$ が表わす文字列の長さを求める。

部分文字列: $\boxed{\text{式}_1}[\boxed{\text{式}_2}..\boxed{\text{式}_3}]$ という式は、 $\boxed{\text{式}_1}$ が表わす文字列の $\boxed{\text{式}_2}$ 番目から $\boxed{\text{式}_3}$ 番目までを取り出した文字列を作る。

空の値: nil は「無い」ことを表わす特別な値である。

配列の作成: `Array.new(式)` という式は、大きさが式 の値であるような配列を作る。作られた配列の中身はすべて nil である。

繰り返し: 変数 の値を 式₁ の値から 式₂ の値まで 1 ずつ順に変化させながら、命令₁ から 命令_n を毎回実行する繰り返しは次のように書く。

```
for 変数 in 式1 .. 式2
  命令1
  ⋮
  命令n
end
```

練習問題確認プログラム

- ex03.rb (練習3.1と3.3-3.9)の結果を以下のアドレスに送れ。

is-komaba@lyon.is.s.u-tokyo.ac.jp

- Subject:欄は、

[is-komaba](#)

としてください。

- 〆切: **11月XX日**

– 練習問題確認プログラムの結果は出席点の一部とします。結果の内容は問いません。(習得状況を知りたい。)

次は何を返す？

```
a = Array.new(2)
```

```
b = Array.new(2)
```

```
for i in 0..1
```

```
  b[i] = a
```

```
  for j in 0..1
```

```
    b[i][j] = i
```

```
  end
```

```
end
```

```
b
```

1. $[[0,0],[0,0]]$

2. $[[0,0],[1,1]]$

3. $[[0,1],[0,1]]$

4. $[[1,1],[1,1]]$

5. $[0,0]$

6. $[0,1]$

7. $[1,1]$

次は何を返す？

```
b = Array.new(2)
```

```
for i in 0..1
```

```
  b[i] = Array.new(2)
```

```
  for j in 0..1
```

```
    b[i][j] = i
```

```
  end
```

```
end
```

```
b
```

1. `[[0,0],[0,0]]`

2. `[[0,0],[1,1]]`

3. `[[0,1],[0,1]]`

4. `[[1,1],[1,1]]`

5. `[0,0]`

6. `[0,1]`

7. `[1,1]`

次は何を返す？

```
a = Array.new(2)
```

```
b = Array.new(2)
```

```
for i in 0..1
```

```
  b[i] = a
```

```
  for j in 0..1
```

```
    b[i][j] = i
```

```
  end
```

```
end
```

```
b
```

次は何を返す？

```
a = Array.new(2)
```

```
b = Array.new(2)
```

```
for i in 0..1
```

```
  b[i] = a
```

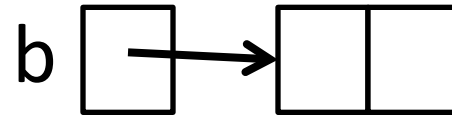
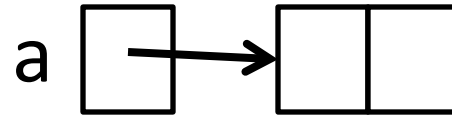
```
  for j in 0..1
```

```
    b[i][j] = i
```

```
  end
```

```
end
```

```
b
```



次は何を返す？

```
a = Array.new(2)
```

```
b = Array.new(2)
```

```
for i in 0..1
```

```
  b[i] = a
```

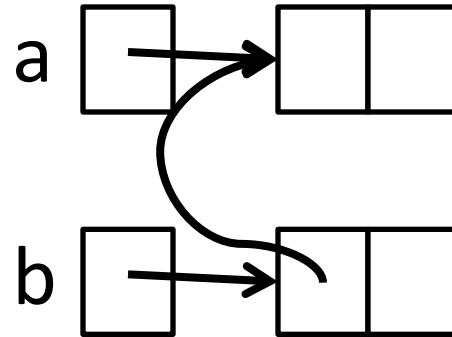
```
  for j in 0..1
```

```
    b[i][j] = i
```

```
  end
```

```
end
```

```
b
```

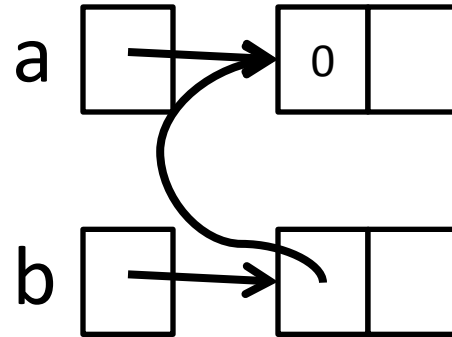


$i = 0$

$b[i] = a$

次は何を返す？

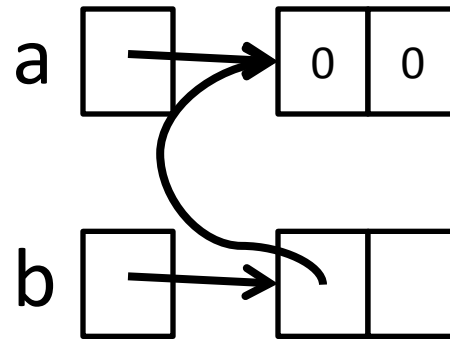
```
a = Array.new(2)
b = Array.new(2)
for i in 0..1
  b[i] = a
  for j in 0..1
    b[i][j] = i
  end
end
b
```



```
i = 0
b[i] = a
j = 0
b[i][j] = i
```

次は何を返す？

```
a = Array.new(2)
b = Array.new(2)
for i in 0..1
  b[i] = a
  for j in 0..1
    b[i][j] = i
  end
end
b
```



```
i = 0
b[i] = a
j = 0
b[i][j] = i
j = 1
b[i][j] = i
```

次は何を返す？

```
a = Array.new(2)
```

```
b = Array.new(2)
```

```
for i in 0..1
```

```
  b[i] = a
```

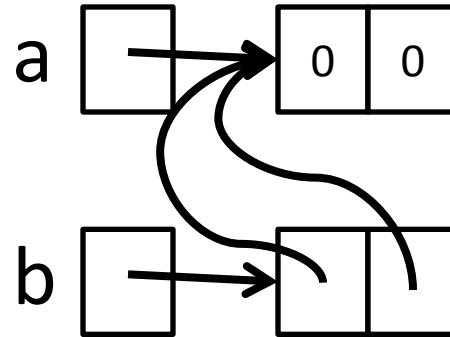
```
  for j in 0..1
```

```
    b[i][j] = i
```

```
  end
```

```
end
```

```
b
```

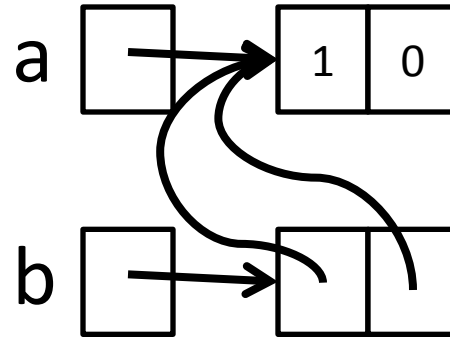


$i = 1$

$b[i] = a$

次は何を返す？

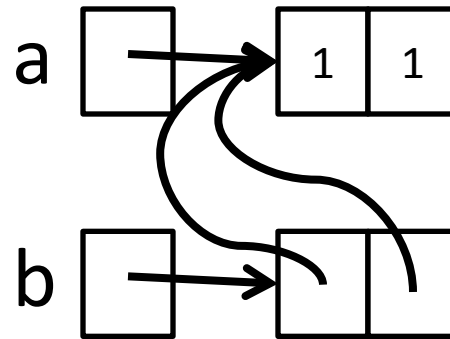
```
a = Array.new(2)
b = Array.new(2)
for i in 0..1
  b[i] = a
  for j in 0..1
    b[i][j] = i
  end
end
b
```



```
i = 1
b[i] = a
j = 0
b[i][j] = i
```

次は何を返す？

```
a = Array.new(2)
b = Array.new(2)
for i in 0..1
  b[i] = a
  for j in 0..1
    b[i][j] = i
  end
end
b
```



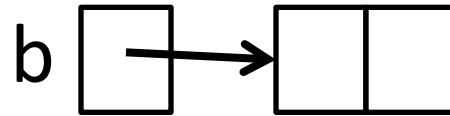
```
i = 0
b[i] = a
j = 0
b[i][j] = i
j = 1
b[i][j] = i
```

次は何を返す？

```
b = Array.new(2)
for i in 0..1
  b[i] = Array.new(2)
  for j in 0..1
    b[i][j] = i
  end
end
b
```

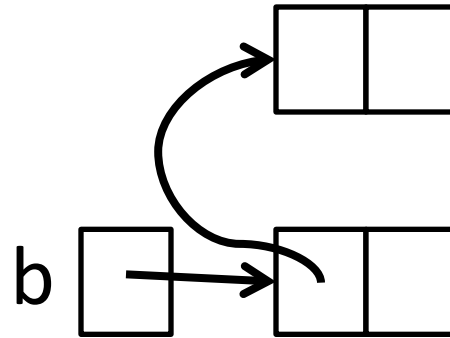
次は何を返す？

```
b = Array.new(2)
for i in 0..1
  b[i] = Array.new(2)
  for j in 0..1
    b[i][j] = i
  end
end
b
```



次は何を返す？

```
b = Array.new(2)
for i in 0..1
  b[i] = Array.new(2)
  for j in 0..1
    b[i][j] = i
  end
end
end
b
```

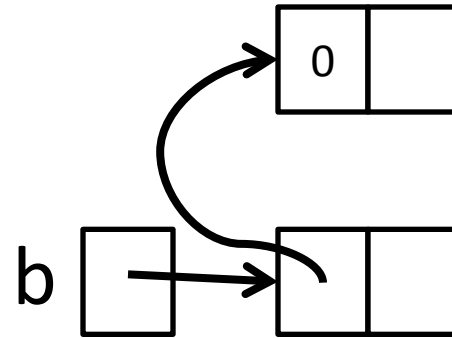


$i = 0$

$b[i] = \text{Array.new}(2)$

次は何を返す？

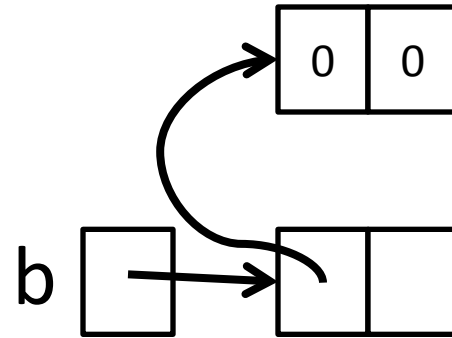
```
b = Array.new(2)
for i in 0..1
  b[i] = Array.new(2)
  for j in 0..1
    b[i][j] = i
  end
end
b
```



```
i = 0
b[i] = Array.new(2)
j = 0
b[i][j] = i
```

次は何を返す？

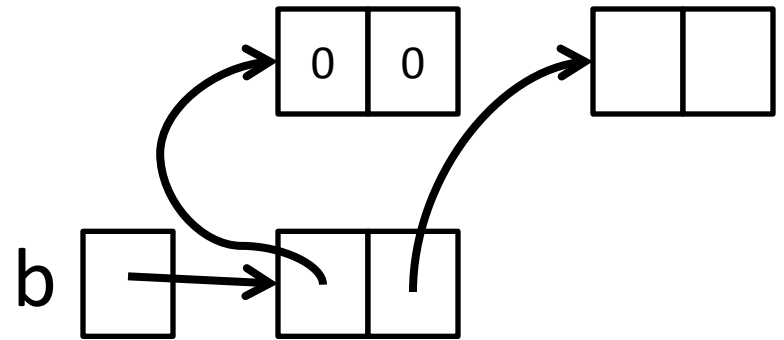
```
b = Array.new(2)
for i in 0..1
  b[i] = Array.new(2)
  for j in 0..1
    b[i][j] = i
  end
end
b
```



```
i = 0
b[i] = Array.new(2)
j = 0
b[i][j] = i
j = 1
b[i][j] = i
```

次は何を返す？

```
b = Array.new(2)
for i in 0..1
  b[i] = Array.new(2)
  for j in 0..1
    b[i][j] = i
  end
end
b
```

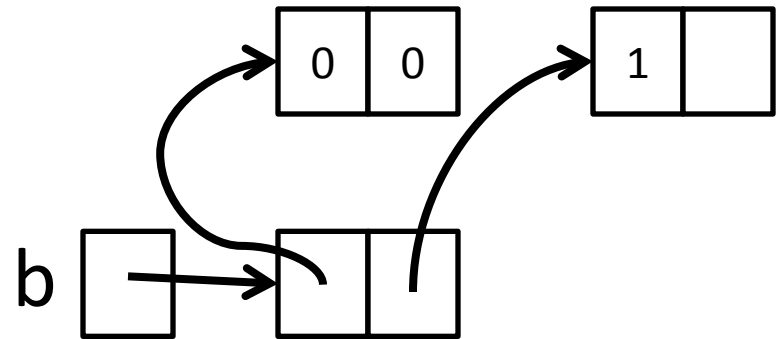


$i = 1$

$b[i] = \text{Array.new}(2)$

次は何を返す？

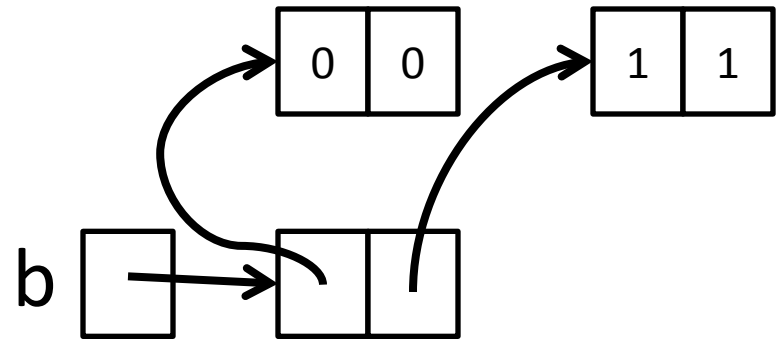
```
b = Array.new(2)
for i in 0..1
  b[i] = Array.new(2)
  for j in 0..1
    b[i][j] = i
  end
end
b
```



```
i = 1
b[i] = Array.new(2)
j = 0
b[i][j] = i
```

次は何を返す？

```
b = Array.new(2)
for i in 0..1
  b[i] = Array.new(2)
  for j in 0..1
    b[i][j] = i
  end
end
b
```



```
i = 1
b[i] = Array.new(2)
j = 0
b[i][j] = i
j = 1
b[i][j] = i
```